

LAPORAN PRAKTIKUM STRUKTUR DATA
BINARY TREE PADA JAVA



DOSEN PENGAMPU :

Dr. Wahyudi, M.T.

OLEH :

RIFKI YULIANDRA

NIM.2311532011

DEPARTEMEN INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS ANDALAS
PADANG

2024

Binary Tree pada Java

I. TUJUAN

- 1.1. Memahami apa yang dimaksud dengan Binary Tree.
- 1.2. Memahami cara membuat program untuk nodenya
- 1.3. Memahami cara membuat program membentuk binary tree tersebut.

II. TEORI

Pohon biner merupakan struktur data fundamental dalam ilmu komputer yang juga berperan penting dalam pemrograman Java. Struktur ini tersusun dari node-node yang menyimpan data, dengan maksimal dua anak untuk setiap node. Setiap anak dibedakan sebagai anak kiri dan anak kanan, sehingga tercipta hubungan hierarkis antar node. Node yang tidak memiliki anak disebut sebagai leaf (daun).

Konsep pohon biner menawarkan beberapa keuntungan. Salah satu kelebihan utamanya adalah efisiensi dalam pencarian data. Berbekal sifat relasional yang dimiliki setiap node (misal: data pada node tertentu selalu lebih besar dari anak kirinya dan lebih kecil dari anak kanannya), kita dapat dengan cepat menelusuri struktur pohon untuk menemukan informasi yang diinginkan. Inilah yang menjadi prinsip dasar dari algoritma pencarian pohon biner.

Java tidak menyediakan implementasi bawaan untuk struktur pohon biner. Namun, kita dapat membangunnya menggunakan class atau interface. Node pada pohon biner umumnya memiliki field untuk menyimpan data dan referensi ke anak kiri dan kanan. Pendefinisian method seperti isEmpty untuk mengecek apakah pohon kosong dan getHeight untuk mendapatkan tinggi pohon dapat mempermudah penggunaan struktur ini dalam program.

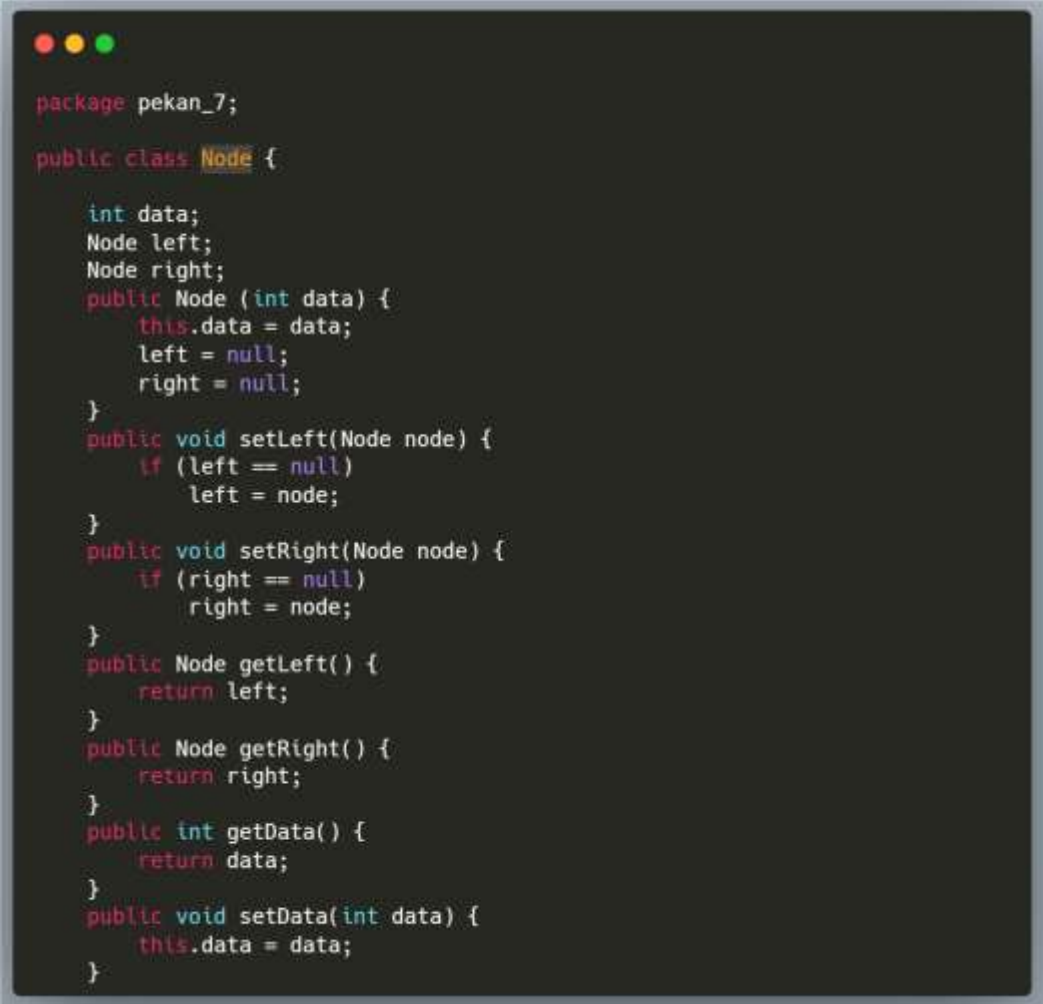
Pohon biner memiliki berbagai macam variasi, masing-masing dengan karakteristik tersendiri. Contohnya adalah pohon pencarian biner (binary search tree) yang di dalamnya data pada setiap node memiliki hubungan tertentu dengan node-node turunannya, sehingga proses pencarian menjadi lebih efisien. Variasi lain seperti pohon AVL dan pohon Red-Black menambahkan mekanisme penyeimbangan untuk memastikan struktur pohon tetap optimal dan menghindari pencarian yang memakan waktu lama.

Secara keseluruhan, pohon biner menjadi struktur data yang sangat berguna dalam berbagai macam permasalahan pemrograman Java. Fleksibilitasnya memungkinkan implementasi berbagai algoritma pencarian dan sorting yang efisien. Penguasaan konsep

pohon biner beserta variasinya akan menjadi bekal yang berharga untuk menghadapi permasalahan pengelolaan data yang kompleks di masa mendatang.

III. LANGKAH KERJA PRAKTIKUM

3.1. Membuat class Node (simpul)



```
package pekan_7;

public class Node {

    int data;
    Node left;
    Node right;
    public Node (int data) {
        this.data = data;
        left = null;
        right = null;
    }
    public void setLeft(Node node) {
        if (left == null)
            left = node;
    }
    public void setRight(Node node) {
        if (right == null)
            right = node;
    }
    public Node getLeft() {
        return left;
    }
    public Node getRight() {
        return right;
    }
    public int getData() {
        return data;
    }
    public void setData(int data) {
        this.data = data;
    }
}
```

```

void printPreorder(Node node) {
    if (node == null)
        return;
    System.out.print(node.data + " ");
    printPreorder(node.left);
    printPreorder(node.right);
}
void printPostorder(Node node) {
    if (node == null)
        return;
    printPostorder(node.left);
    printPostorder(node.right);
    System.out.print(node.data + " ");
}
void printInorder(Node node) {
    if (node == null)
        return;
    printPreorder(node.left);
    System.out.print(node.data + " ");
    printPreorder(node.right);
}
public String print() {
    return this.print("", true, "");
}
public String print(String prefix, boolean isTail, String sb) {
    if (right != null) {
        right.print(prefix + (isTail ? "| " : " "), false, sb);
    }
    System.out.println(prefix + (isTail ? "\\-- " : "/-- ") + data);
    if (left != null) {
        left.print(prefix + (isTail ? " " : "| "), true, sb);
    }
    return sb;
}
}

```

Kelas Node merepresentasikan sebuah node dalam pohon biner. Setiap node memiliki tiga atribut:

1. **data**: nilai integer yang disimpan dalam node
2. **left**: referensi ke node anak kiri
3. **right**: referensi ke node anak kanan

Kelas Node juga memiliki beberapa metode:

- Node(int data): konstruktor yang membuat node baru dengan nilai data
- setLeft(Node node): mengatur node anak kiri
- setRight(Node node): mengatur node anak kanan
- getLeft(): mengembalikan node anak kiri
- getRight(): mengembalikan node anak kanan
- getData(): mengembalikan nilai data dalam node

- `setData(int data)`: mengatur nilai data dalam node

Metode Pencetakan

Kelas Node juga memiliki tiga metode untuk mencetak pohon biner dalam urutan yang berbeda:

- `printPreorder(Node node)`: mencetak pohon biner dalam urutan preorder (root-left-right)
- `printPostorder(Node node)`: mencetak pohon biner dalam urutan postorder (left-right-root)
- `printInorder(Node node)`: mencetak pohon biner dalam urutan inorder (left-root-right)

Metode Pencetakan Grafis

Metode `print()` dan `print(String prefix, boolean isTail, String sb)` digunakan untuk mencetak pohon biner dalam bentuk grafis. Metode ini menggunakan teknik rekursif untuk mencetak setiap node dalam pohon biner.

Metode `print()` memanggil metode `print(String prefix, boolean isTail, String sb)` dengan parameter awal. Metode `print(String prefix, boolean isTail, String sb)` mencetak setiap node dalam pohon biner dengan menggunakan prefix dan simbol untuk mengindikasikan hubungan antar node.

Dalam metode `print(String prefix, boolean isTail, String sb)`, jika node memiliki anak kanan, maka metode ini akan memanggil dirinya sendiri dengan parameter yang berbeda untuk mencetak anak kanan. Kemudian, metode ini akan mencetak nilai **data** dalam node dan akhirnya memanggil dirinya sendiri untuk mencetak anak kiri.

3.2. Membuat class Btree

```
package pekan_7;

public class BTree {
    private Node root;
    private Node currentNode;
    public BTree(){
        root = null;
    }
    public boolean search(int data) {
        return search(root, data);
    }
    private boolean search(Node node, int data) {
        if (node.getData() == data)
            return true;
        if (node.getLeft() != null)
            if (search(node.getLeft(), data))
                return true;
        if (node.getRight() != null)
            if (search(node.getRight(), data))
                return true;
        return false;
    }
    public void printInorder() {
        root.printInorder(root);
    }
    public void printPreorder() {
        root.printPreorder(root);
    }
    public void printPostorder() {
        root.printPostorder(root);
    }
    public Node getRoot() {
        return root;
    }
}
```

```

public boolean isEmpty() {
    return root == null;
}
public int countNodes() {
    return countNodes(root);
}
private int countNodes(Node node) {
    int count = 1;
    if (node == null) {
        return 0;
    } else {
        count += countNodes(node.getLeft());
        count += countNodes(node.getRight());
        return count;
    }
}
public void print() {
    root.print();
}

public Node getCurrent() {
    return currentNode;
}

public void setCurrent(Node node) {
    this.currentNode = node;
}

public void setRoot(Node root) {
    this.root = root;
}
}

```

Kelas BTree merepresentasikan sebuah binary tree yang memiliki beberapa atribut dan metode.

a. Atribut

- root: node akar dari binary tree
- currentNode: node yang sedang aktif dalam binary tree

b. Metode

- BTree(): konstruktor yang membuat binary tree baru dengan node akar yang null
- search(int data): mencari node dengan nilai data dalam binary tree dan mengembalikan true jika ditemukan, false jika tidak
- printInorder(), printPreorder(), printPostorder(): mencetak binary tree dalam urutan inorder, preorder, dan postorder
- getRoot(): mengembalikan node akar dari binary tree
- isEmpty(): mengembalikan true jika binary tree kosong, false jika tidak

- `countNodes()`: menghitung jumlah node dalam binary tree
- `print()`: mencetak binary tree dalam bentuk grafis
- `getCurrent()`: mengembalikan node yang sedang aktif dalam binary tree
- `setCurrent(Node node)`: mengatur node yang sedang aktif dalam binary tree
- `setRoot(Node root)`: mengatur node akar dari binary tree

c. Metode Search

Metode `search(int data)` menggunakan teknik rekursif untuk mencari node dengan nilai data dalam binary tree. Metode ini memanggil dirinya sendiri untuk mencari node dalam subtree kiri dan kanan.

d. Metode Print

Metode `printInorder()`, `printPreorder()`, dan `printPostorder()` menggunakan metode yang sama seperti dalam kelas `Node` untuk mencetak binary tree dalam urutan yang berbeda.

e. Metode Count Nodes

Metode `countNodes()` menggunakan teknik rekursif untuk menghitung jumlah node dalam binary tree. Metode ini memanggil dirinya sendiri untuk menghitung node dalam subtree kiri dan kanan.

f. Metode Print Grafis

Metode `print()` menggunakan metode yang sama seperti dalam kelas `Node` untuk mencetak binary tree dalam bentuk grafis.

3.3. Membuat class TreeMain untuk mengeksekusi kedua program sebelumnya

```
package pekan_7;

public class TreeMain {

    public static void main(String[] args) {
        // Membuat Pohon
        BTree tree = new BTree();
        System.out.print("Jumlah simpul pohon: ");
        System.out.println(tree.countNodes());

        //Menambahkan simpul data 1
        Node root = new Node(1);

        // Menjadikan simpul 1 sebagai root
        tree.setRoot(root);
        System.out.println("Jumlah simpul jika hanya ada root");
        System.out.println(tree.countNodes());
        Node node2 = new Node(2);
        Node node3 = new Node(3);
        Node node4 = new Node(4);
        Node node5 = new Node(5);
        Node node6 = new Node(6);
        Node node7 = new Node(7);
        root.setLeft(node2);
        node2.setLeft(node4);
        node2.setRight(node5);
        node5.setLeft(node7);
        root.setRight(node3);
        node3.setRight(node6);
    }
}
```

```
        // Set root
        tree.setCurrent(tree.getRoot());
        System.out.println("menampilkan simpul terakhir: ");
        System.out.println(tree.getCurrent().getData());
        System.out.println("Jumlah simpul; setelah simpul 7 ditambahkan");
        System.out.println(tree.countNodes());
        System.out.println("InOrder: ");
        tree.printInorder();
        System.out.println("\nPreOrder: ");
        tree.printPreorder();
        System.out.println("\nPostOrder: ");
        tree.printPostorder();
        System.out.println("\nMenampilkan simpul dalam bentuk pohon");
        tree.print();
    }
}
```

Program ini membuat sebuah objek BTree bernama tree dan mencetak jumlah simpul pohon yang awalnya kosong. Adapun kerja program tersebut adalah :

- a. Menambahkan simpul dengan membuat beberapa objek Node dengan nilai 1, 2, 3, 4, 5, 6, dan 7. Kemudian, program ini mengatur struktur pohon dengan menghubungkan simpul-simpul tersebut.
- b. Mengatur root, yaitu mengatur simpul dengan nilai 1 sebagai root pohon dan mencetak jumlah simpul pohon setelah itu.
- c. Menambahkan simpul lainnya ke dalam pohon dan mengatur struktur pohonnya.
- d. Mencetak simpul terakhir dengan mencetak nilai simpul terakhir yang ditambahkan, yaitu simpul dengan nilai 7.
- e. Mencetak jumlah simpul pohon setelah semua simpul ditambahkan.
- f. Mencetak Pohon dalam Berbagai Urutan, yaitu inorder, preorder, dan postorder menggunakan metode printInorder(), printPreorder(), dan printPostorder().
- g. Mencetak Pohon dalam Bentuk Grafis menggunakan metode print().

Dan adapun output yang dihasilkan dari program-program tersebut:

```
<terminated> TreeMain [Java Application] C:\Users\User\p2\pool\p
Jumlah simpul pohon: 0
Jumlah simpul jika hanya ada root
1
menampilkan simpul terakhir:
1
Jumlah simpul; setelah simpul 7 ditambahkan
7
InOrder:
4 2 7 5 1 3 6
PreOrder:
1 2 4 5 7 3 6
PostOrder:
4 7 5 2 6 3 1
Dmenampilkan simpul dalam bentuk pohon
|          /-- 6
|      /-- 3
|-- 1
|      /-- 5
|      |      \-- 7
|-- 2
|      \-- 4
```

Dalam program ini, kita dapat melihat bagaimana kelas BTree dan Node digunakan untuk membuat dan mengelola pohon, serta mencetaknya dalam berbagai urutan dan bentuk grafis.

IV. KESIMPULAN

Ketiga program yang dibahas merupakan implementasi struktur data pohon biner pada bahasa pemrograman Java. Program pertama adalah kelas Node yang mewakili sebuah node dalam pohon biner, dengan data, properti kiri dan kanan, serta metode untuk mengatur dan mengambil nilai properti tersebut. Program ini juga memiliki metode untuk mencetak pohon biner secara pre-order, in-order, dan post-order.

Program kedua adalah kelas BTree yang mewakili pohon biner, dengan properti root dan currentNode, serta metode untuk mengatur dan mengambil nilai properti tersebut. Program ini juga memiliki metode untuk mencari node dengan nilai tertentu, menghitung jumlah node, dan mencetak pohon biner secara order, preorder, dan postorder. Selain itu, program ini juga memiliki metode untuk mencetak pohon biner secara grafis.

Program ketiga adalah contoh penggunaan kelas BTree dan Node untuk membuat dan mengelola pohon biner. Program ini membuat objek BTree dan menambahkan beberapa node pada pohon, kemudian mencetak jumlah node, mencetak pohon secara berurutan, preorder dan postorder, serta mencetak pohon dalam bentuk grafik. Jadi, program ini menunjukkan bagaimana kelas BTree dan Node dapat digunakan untuk membuat dan mengelola pohon biner.

Secara keseluruhan, ketiga program ini menunjukkan cara mengimplementasikan struktur data pohon biner dalam bahasa pemrograman Java. Dengan menggunakan kelas Node dan BTree, kita dapat dengan mudah membuat dan mengelola pohon biner dan mencetaknya dalam berbagai urutan dan bentuk grafik. Ini berguna dalam berbagai aplikasi, seperti pengelolaan data, penelitian data, dan visualisasi data.